الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التكوين والتعليم المهنيين
Ministère de la Formation et de l'Enseignement Professionnels

Centre National de l'Enseignement Professionnel à Distance (CNEPD)

المركز الوطني للتعليم المهني عن بعد
(م.و.ت.م.ب)

# COURS DE STRUCTURE MACHINE

## SÉRIE 02

**OBJECTIF PÉDAGOGIQUE :**

À l'issue de cette série, vous serez capable de présenter l'information et notion du codage ainsi, d'identifier les différents types de représentation des données en mémoire.

**PLAN DE LA LEÇON:**

**INTRODUCTION**

**I- L'INFORMATION DIGITALE**

1- Information digitale
2- Notion de codage
3- Taille de l'information

**II- LA REPRÉSENTATION DES DONNÉES EN MÉMOIRE**

1- Représentation des nombres
2- Représentation des informations non numériques
3- Les différents types de codage

**EXERCICES D'APPLICATION**

**CORRECTION DES EXERCICES**

## INTRODUCTION :

Toute réalité descriptible peut être représentée par une abstraction du système d'information. En effet, toute description humaine utilise un langage, et tous les langages sont représentables par des signes qui peuvent être codifiés sous une forme finie.

La théorie des langages nous apprend que ces langages :

- Peuvent être finis, c'est à dire disposent d'un certain nombre des sémantiques qu'on ne peut étendre.
- Sont infinis, et peuvent être ramenés à une forme finie (par exemple un alphabet).

Encoder l'information, c'est la représentation avec un code. Un code est un ensemble de symboles qui permet de représenter d'une manière «univoque», «transportable», et «opérable» des significations.

L'ordinateur est une machine à dimension finie. Elle a fort intérêt à manœuvrer des encodages finis.

Un encodage fini propose un jeu de symbole particulier à un nombre représentable sous sa forme la plus élémentaire dans la machine : Un nombre binaire.

# I- L'INFORMATION DIGITALE :

## 1- Information digitale :

Une information numérique (en anglais «digital») est une information ayant été quantifiée et échantillonnée, par opposition à une information dite «analogique» qui est une information «brute», à priori non quantifiée ni échantillonnée, ou bien pouvant être perçue comme "quantifiée et échantillonnée à l'infini". Le terme «numérique» est surtout employé en informatique et en électronique, notamment pour le son, la photographie, la vidéo, le cinéma. Ainsi, L'information digitale élémentaire est alternative : Il y a ou il n'y a pas de courant dans un fil électrique.

Conventionnellement ces états sont notés 1 et 0, l'information contenue est appelée digit ou bit.

## 2- Notion de codage :

Comment mettre en mémoire des objets différents (instructions, nombres, texte,…) ?

- Pour chaque type d'objet, on choisit un code.
- Un code est une règle pour faire correspondre un nombre (sa représentation) à chaque objet du type.

- Le codage consiste à établir une loi de correspondance appelée code entre les informations à représenter et les configurations binaires.

- Vers la fin des années 30, Claude Shannon démontra qu'à l'aide de « contacteurs » (interrupteurs) fermés pour « vrai » et ouvert pour « faux » il était possible d'effectuer des opérations logiques en associant le nombre 1 pour « vrai » et 0 pour « faux » .

- Ce codage de l'information est nommé base binaire. C'est avec ce dernier que fonctionnent les ordinateurs. Il consiste à utiliser deux états (représentés par le chiffre 0 et 1) pour coder les informations.

Pour coder les caractères (lettre de l'alphabet, signes de ponctuation, chiffres...), le premier code utilisé, le code ASCII (American **S**tandard Coding for Informatic Interchnage) a été standardisé sur 7 bits permettant de représenter 128 signes. Il ne permet pas la représentation des lettres accentuées ou autres signes spécifiques. Pour cela, il a été étendu sur 8 bits avec des interprétations différentes suivants les systèmes (le code ASCII du macintosh et du PC ne sont pas les mêmes…).

Afin de rendre les systèmes interopérables et de tenir compte de l'ensemble des signes des différentes langues, une autre représentation a vu le jour dans l'UNICODE ;

- Le codage des informations consiste à transcrire les messages d'une source d'information ;
- Sous la forme d'une suite de caractères pris dans un alphabet prédéfini. Les objectifs du codage sont principalement de quatre ordres **:**

- De transcrire les informations sous une forme permettant d'élaborer assez facilement le signal qui supportera les informations, ou de manipuler aisément ces informations automatiquement. Pour cela, différents codes de représentation de l'information sont utilisés en fonction des différentes applications envisagées et on utilise assez souvent des opérations de transcodage ;
- De réduire la quantité des symboles d'information (en terme de nombre total de symboles utilisés) nécessaires à la représentation des informations : c'est un rôle économique ;
- De lutter contre les dégradations (distorsions, bruit) amenées par le canal de transmission et conduisant à des erreurs de reconstruction de l'information en sortie du canal de transmission (en réception) ;
- D'assurer un secret dans la transmission des informations de façon à rendre l'information inintelligible sauf au destinataire de celle-ci.

**Exemples de codes :**

- Le code des mois : Janvier = 1, Février = 2, … Décembre = 12.
- C'est arbitraire (pourquoi commencer en janvier ?)
- Si tout le monde utilise le même code, tout le monde comprend que "9" veut dire "Septembre".

### 3- Taille de l'information :

- Toute information dans un système informatique est représentée sous la forme d'un paquet de bits ;
- La différence entre un type d'information et un autre est donnée seulement par le contexte: La même séquence de bits peut représenter un nombre entier, un nombre réel, un caractère, une instruction, un son, ……..etc.

**Information = bits + contexte**

- On appelle généralement "taille d'un mot" le nombre de bits utilisés par un ordinateur pour stocker un entier ;
- La plupart des ordinateurs possèdent des mots 32 bits, bien que la tendance est d'aller vers les 64 bits ;
- Les ordinateurs peuvent travailler sur plusieurs formats de données, fractions ou multiples de la taille du mot. Toutefois, la taille de tous ces formats est toujours un multiple d'un byte.
- Les données sont stockées dans une mémoire, chaque donnée à une adresse différente ;
- Chaque byte dans une mémoire possède une adresse différente. Si une donnée contient plus d'un byte, l'adresse de la donnée correspond à celle du premier byte.

## Caractère :

- Caractère = Lettre (**a**, **b**,.., **A**, **B**,..), chiffre (**1**,…,**9**), ponctuation (, ; : . ! ? {}[]), …..Etc. Comment les noter en mémoire ???

- Code inventé par l'association des ingénieurs américains (ascii).
- Caractères standard sur 7 bits (sans accents), étendu sur 8 bits.
  ⇒ Chaque caractère étendu est codé par un nombre entre 1 et 255 (standard : de 1 à 127).

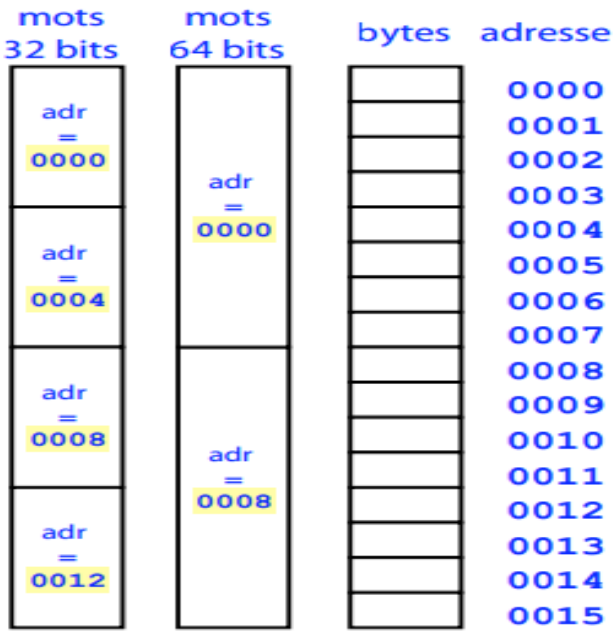
## Mot :

Un mot est le nombre maximum de bits que le processeur est capable de lire ou d'écrire en une fois :

- Un mot de 8 bits s'appelle 1 octet ;
- Les processeurs courants ont des mots de 4 ou 8 octets. Ils lisent aussi, à l'occasion 1 ou 4 octets à la fois.
- La taille du mot mémoire est un multiple de 8 bits.

On rencontre des ordinateurs ayant des mots mémoire de :

- 4 bits (historique: Intel 4004)
- 8, 16 bits (autrefois)
- 32, ou 64 bits (aujourd'hui)




-

Une adresse est stockée dans un mot de la mémoire d'un ordinateur ;

- Le nombre de bits d'un mot limite donc la taille maximale de la mémoire d'un ordinateur ;
- Si un ordinateur utilise des mots de 32 bits, la taille maximale de sa mémoire est de $2^{32}$ bytes, c'est-à-dire 4 gigabytes:
  - $2^{32}=2^2*2^{30}=4GB$
- Les différents bytes d'un mot peuvent être ordonnés de différentes façons dans la mémoire ;
- Les deux ordonnancements les plus utilisés sont:

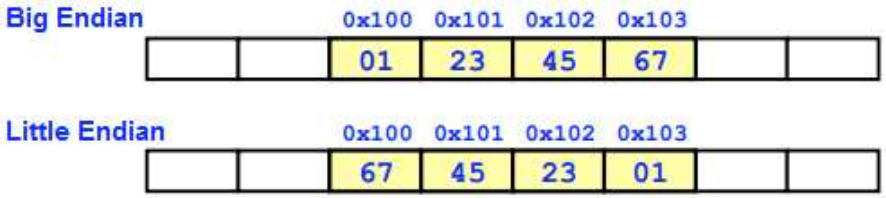
**Big Endian :**

- Le byte de poids fort est mis à l'adresse inférieure (Le mot commence par le byte de poids fort).
- Utilisé par les ordinateurs Sun et Macintosh, par exemple.

**Litlle Endian :**

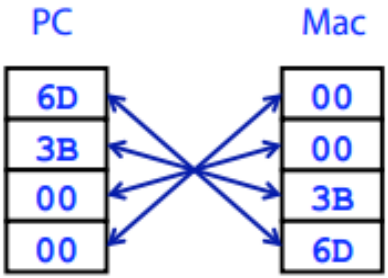
- Le byte de poids faible est mis à l'adresse inférieure (le mot commence par le byte de poids faible).
- Utilisé par les ordinateurs PC et Alpha, par exemple :

Supposez que la valeur de la variable toto est **0x01234567** et qu'elle est stockée à l'adresse **0 x 0100** (c'est-à-dire **toto = 0 x 0100**).




**Exemple**, sur un ordinateur **32 bits**:
Int toto = 15213;

Décimal : 15213
Binaire : 0011 1011 0110 1101
Hexadécimal : 3 B 6 D




Bien que, pour la plupart des cas, l'ordonnancement des bytes est transparent pour l'utilisateur, il existe des situations où il peut être à l'origine des erreurs.

- Lors de la transmission de données entre deux ordinateurs : Le protocole de communication doit spécifier l'ordre de transmission des bytes.
- Lors de l'examen d'un programme en assembleur (Debugging, par exemple).
- Lors du traitement des données à bas niveau, possible avec des langages Tels que C.

## II- REPRÉSENTATION DES DONNÉES EN MÉMOIRE :

### 1- Représentation des nombres :

#### 1.1 Représentation des nombres négatifs :

La vie n'étant pas faite que de choses positives, il a fallu introduire les nombres négatifs .Leur représentation est réalisée grâce au bit de poids fort on parle également d'entiers signés :

- nombres positifs → Le bit de poids fort est **0**
- nombre négatifs → Le bit de poids fort est **1**

Partant de ce principe, il existe différentes technique pour représenter les nombres négatifs :

➢ **Représentation par signe et valeur absolue :**

Le nombre négatif est codé de la manière suivante :

- Le bit de poids fort est à **1**.
- Les autres bits contiennent la valeur absolue du nombre.

**Exemple sur un octet :**

| Décimal | Binaire | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **12** | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| **-12** | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

Un inconvénient de cette méthode est que pour réaliser des opérations entre nombres signés, il faut faire un traitement particulier du bit de signe. Le résultat final ne pouvant être obtenu de façon simple (addition ou soustraction des deux nombres).

#### 1.2 Représentation des nombres en virgule fixe :

Dans cette représentation, la position de la virgule est laissée à l'appréciation du programmeur. En fait, il s'agit uniquement d'un format d'affichage.

**Exemple :**

Les prix sur une facture sont représentés systématiquement avec 2chifres derrière la virgule 12,50€ sera représenté comme 1250.

- Dans cette représentation, les nombre décimaux sont représentés comme des entiers dans l'ordinateur ;
- Le programmeur sait que les chiffres ont 2 chiffres derrière la virgule ;
- Il lui appartient de placer la virgule lors de l'affichage ;
- De la même manière, il placera le sigle derrière le montant.

### 1.3 Représentation des nombres flottants :

La représentation en format fixe présente l'inconvénient majeur de sa taille, à savoir le nombre est limité .Pour passer cette limitation, une présentation existe à savoir la représentation en virgule flottante.

**Codage virgule flottante :**

Chaque nombre réel peut s'écrire de la façon suivante :

$$N = \pm M * b^e$$

**M : mantisse, b : la base, e : l'exposant**

Les données sont composées de deux parties :

- Exposant : La partie variable représente la puissance de **10** de la partie fixe ;
- Mantisse : La partie fixe représentant les chiffres.

**Exemple :**

$123{,}45\ 10^3$ → 123,45 la mantisse
→ $10^3$ l'exposant

Une représentation en virgule flottante n'est pas unique, on pourrait aussi, écrire :

- $123{,}5\ 10^2$
- 123450 ici l'exposant est 0
- $0{,}12345\ 10^6$

Il faut donc, les représenter sous forme normalisée afin que la représentation ne varie pas d'un logiciel à un autre.

### a- La forme normalisée :

Un nombre normalisé en virgule flottante est un nombre dans lequel le chiffre qui précède la virgule est un **0** et le chiffre qui suit la virgule n'est pas un **0**.

Ce **0** est omis dans l'écriture de la mantisse et le nombre est stocké comme un entier :

onc : Si on veut représenter $123{,}45\ 10^3$ :

- $0{,}01235\ 10^7$ n'est pas normalisé
- $0{,}12345\ 10^6$ est normalisé et est stocké comme :
  - 123455 comme mantisse
  - 6 comme exposant

### b- Le codage dans la machine :

L'écriture en virgule flottante est normalisée par **L'IEE.** La forme d'écriture sur **32 bits** (simple précision) prendra ce format :

| Signe exposant | Exposant | Signe mantisse | Mantisse |
|---|---|---|---|
| 1 bit | 7 bit | 1 bit | 23 bit |

**Exemple :** On codera 123456 comme mantisse → 1E240
On codera 6 comme exposant → 6

Le nombre s'écrit alors :

| Exposant | | | | | | | | Mantisse | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | | | | | | | | 1 | | | | | | | | E2 | | | | | | | | 40 | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

### 1.4 Le complément à « un » :

Le complément à 1 (noté **C1**) est également appelé complément logique, il consiste à inverser chaque bit **(0→1 et 1→0)**

- L'entier positif est représenté sous sa forme naturelle
- L'entier négatif est représenté par le complément à 1

**Exemple sur « un octet » :**

| Décimal | Binaire | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 12 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| -12 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |

### 1.5 Le complément à « deux » :

Le complément à 2 (noté **C2**) est également appelé complément vrai, il consiste à ajouter **1** en binaire au complément a un.

- L'entier positif est représenté en binaire naturel.
- L'entier négatif est représenté par le complément à **2** de son opposé.

**Exemple sur « un octet » :**

| Décimal | Binaire | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 12 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| C1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| +1 | | | | | | | | 1 |
| -12 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |

A partir de ce moment, il devient aisé de réaliser des opérations sur des nombres signés, une soustraction **(a-b)** devenant une addition **(a+(-b)) (-b)** étant le complément à **2** de **b**.

**Exemple :** (17,-12)

| Décimal | Binaire | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 17 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| -12 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

**Le résultat se lit directement :**

- Si le bit de poids fort est **0**, le résultat est positif.
- Si le bit de poids fort est **1**, le résultat est négatif.

**Exemple :** (5,12)

| Décimal | Binaire | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| -12 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |

Ici, le résultat est négatif pour en connaitre la valeur absolue, il suffit de faire le complément à 2 du résultat :

| Décimal | Binaire | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| C1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| +1 | | | | | | | | 1 |
| | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

La valeur absolue du résultat est **0000 000111$_2$→7$_{10}$**

**5 - 12 = -7 CQFD**

## 2- Représentation des informations non numériques :

### 2.1- Représentation ou codage des caractères :

Les caractères englobent : Les lettres alphabétiques (A, a, B,..), les chiffres, et les autres symboles (>, / : ….).

Le codage le plus utilisé est **l'ASCII (American Standard Code for Information Inter change)**. Dans ce codage chaque caractère est représenté sur 8 bits.

Avec **8 bits** on peut avoir $\mathbf{2^8 = 256}$ **combinaisons**.

Chaque combinaison représente un caractère.

**Exemple :**

- Le code 65 $(01000001)_2$correspond au caractère **« A »**
- Le code 97 (01100001) correspond au caractère **« a »**
- Le code 58 (00111010) correspond au caractère **« : »**

Actuellement il existe un autre code sur 16 bits, se code s'appelle **UNICODE**.

### 2.2- Représentation ou codage des instructions :

Une instruction est la réunion des différents signaux de commande du chemin de données du processeur.

- Une instruction est découpée en champs.
- Un code opération **((OPC od))**.
- Des informations complémentaires sur l'emplacement des données sources et de la destination.
- La technique associée à la localisation on des opérandes d'une instruction s'appelle l'adressage ou le mode d'adressage.

➢ **Format générale d'une instruction :**

- Une instruction désigne un ordre (minimal) donné au processeur.
- Au contrôleur de savoir comment répondre à cet ordre : Décodage en commandes.

- Code Op sur m bits = addition, multiplication, rangement…

➢ **Les choix de codage :**

Le codage des instructions sur **« m »** bits dépend :

- Du nombre d'opérandes par instruction **(champs de l'instruction)**.
- Du mode d'adressage de ces opérandes **(K bits)**.
- Du nombre d'instruction **(m bits)**.
- Du nombre de registres de l'architecture.

Les compromis de l'architecte :

- Le désir d'avoir autant de registres que possible.
- L'impact sur la taille moyenne des instructions.
- La facilité d'avoir des instructions de longueurs égales pour le décodage.

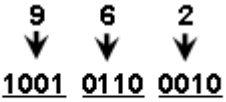
## 3- Les différents types de codage :

### 3.1- BCD : (Binary Coded Décimal) :

Qui peut se traduire en français par décimal codé binaire, est un système de numération utilisé en électronique et en informatique pour coder des nombres d'une façon relativement proche de la représentation humaine usuelle **(en base 10)**. En **BCD**, les nombres sont représentés en chiffres décimaux et chacun de ces chiffres est codé sur quatre bits .

| Chiffre | Bits | Chiffre | Bits |
|---|---|---|---|
| **0** | 0000 | 5 | 0101 |
| **1** | 0001 | 6 | 0110 |
| **2** | 0010 | 7 | 0111 |
| **3** | 0011 | 8 | 1000 |
| **4** | 0100 | 9 | 1001 |

**Exemple :** Pour convertir un nombre décimal en code **DCB** il suffit de chercher **l'**équivalent de chaque chiffre dans le tableau.




Donc $(962)_{10} = (1001\ 0110\ 0010)_{DCB}$

### 3.1- EXCESS 3 :

Le code décimal binaire **Excess-3 (XS-3)**, aussi appelé représentation biaisée ou **Excess-N**, est un système numérique utilisé sur quelques vieux ordinateurs qui utilisent un nombre **N** prédéfinit comme nombre biaisant. **En XS-3**, les nombres sont représentés comme chiffres décimaux, et chaque chiffre est représenté par quatre bits comme la valeur **BCD** plus **3**.

| Décimal | Binary | Décimal | Binary |
|---|---|---|---|
| **0** | 0011 | 9 | 1100 |
| **1** | 0100 | 8 | 1011 |
| **2** | 0101 | 7 | 1010 |
| **3** | 0110 | 6 | 1001 |
| **4** | 0111 | 5 | 1000 |

Pour encoder un nombre comme **127**, on encode simplement chacun des nombres décimaux ci-dessus, donnant **(0100, 0101, 1010)**.

L'avantage principal de l'encodage **XS-3** sur l'encodage **BCD** est qu'on peut calculer le complément à 9 d'un nombre décimal (pour soustraire) aussi facilement qu'on peut calculer le complément à 1 d'un nombre binaire; simplement en inversant les bits.

**Excess-3** fonctionne aussi sur un algorithme différent de l'encodage **BCD** ou des nombres binaires normaux. Quand vous additionnez deux nombres **XS-3** ensembles, le résultat n'est pas un nombre XS-3. Par exemple, quand vous ajoutez **1** et **0** en **XS-3** la réponse semble être **4** au lieu de **1**. Afin de corriger ce problème, quand vous avez fini d'additionner chaque chiffre, vous devez soustraire **3 (en**

**binaire: 11)** si le chiffre est inférieur au décimal **10** et ajouter **3** si le nombre est supérieur ou égal à **10**.

### 3.2- EBCDIC: Extended Binary Coded Decimal Interchange Code

Est un mode de codage des caractères sur 8 bits créé par IBM à l'époque des cartes perforées. Il existe au moins 6 versions différentes bien documentées (et de nombreuses variantes parfois créées par des concurrents d'IBM), incompatibles entre elles. Ce mode de codage a été critiqué pour cette raison, mais aussi parce que certains caractères de ponctuation ne sont pas disponibles dans certaines versions. Ces disparités ont parfois été interprétées comme un moyen pour IBM de conserver ses clients captifs.

EBCDIC est encore utilisé dans les systèmes AS/400 d'IBM ainsi que sur les mainframes sous MVS, VM ou DOS/VSE.

### 3.3- ASCII : ( American Standard Code for Information Interchange)

La norme **ASCII** est largement utilisée en informatique pour coder les caractères. Ce nom provient de l'acronyme anglais "**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange" qui signifie en français **"Code américain normalisé pour l'échange d'information"**.

L'**ASCII** est une norme de codage la plus répandue et compatible avec le plus de support. Il contient l'ensemble des caractères alphanumérique utilisé en anglais. Initialement codé sur **7 bits** (plus un code de parité), l'ASCII étendu est néanmoins codé sous **8 bits** dans le but d'ajouter des caractères, tel que des caractères spéciaux et les lettres accentuée utilisée en français.

La norme **ASCII** permet ainsi à toutes sortes de machines de **stocker**, **analyser** et **communiquer de l'information textuelle**. En particulier, le quasi totalité des ordinateurs personnels et des stations de travail utilisent l'encodage **ASCII**.

Le codage **ASCII** est souvent complété par des correspondances supplémentaires afin de permettre l'encodage informatique d'autres caractères, comme les caractères accentués par exemple. Cette norme s'appelle **ISO-8859** et se décline par exemple en **ISO-8859-1** lorsqu'elle étend **l'ASCII** avec les caractères accentués d'Europe occidentale.

Il existe d'autres normes que **l'ASCII**, comme **l'Unicode** par exemple, qui présente l'avantage de proposer une version unifiée des différents encodages de caractères complétant **l'ASCII** mais aussi de permettre l'encodage de caractères autres que ceux de l'alphabet latin. Le codage **UTF8** de l'Unicode est une extension **d'ASCII** utilisant le **8$^{e}$ bit**. *(Source Dicodunet.com)*.

- **Transmissions en mode ASCII / Binaire :**

Il existe deux modes de transmission des fichiers Informatiques : Le mode **ASCII** et le mode **Binaire**. Un mauvais choix dans le mode de transmission peut rendre un fichier inexploitable.

En mode **ASCII**, le logiciel de transmission adressera le code **ASCII** de chaque caractère. Ce mode est particulièrement destiné à la transmission des fichiers dits "texte" (**HTML**, sources, scripts).

En mode **Binaire**, le logiciel de transmission enverra les bits par paquets. Ce mode convient aux fichiers, comme les exécutables, les images, les sons, ….etc.

Le choix du mode **ASCII** / **binaire** est souvent laissé à l'appréciation du logiciel de transmission qui déterminera automatiquement le mode approprié en fonction de l'extension ou du contenu du fichier à transmettre.

Certaines transmissions sont effectuées en mode **hexadécimal**. Ce mode de transmission permet de transmettre des fichiers **binaires** en mode **ASCII,** c'est à dire en n'utilisant que les 128 premiers caractères de la Table **ASCII**.

## EXERCICES D'APPLICATION :

**1-** Calculer 20-30 et 48-79 ?

**2-** Donner la représentation des deux nombres N1= $(-0,014)_8$ et N2=$(0,14)_8$ sur la machine Suivante :

| Signe mantisse | Exposant biaisé (décalé) | Mantisse normalisée |
|---|---|---|
| **1 bit** | **5 bits** | **10 bits** |

**3-** Calculer N2-N1 ?

## CORRECTION DES EXERCICES :

**1-** Le calcul :

- 20-30 :

  20→ 0001 0100

  30→ 0001 1110(-30) → 1110 0010

  1111 0110 le résultat est négatif

  → 0000 1010(en valeur absolue)

- 48-79 :

  00110000 –(01001111) → 00110000 + 10110001 = 11100001 →$-31_{10}$

**2-** Avant de représenter les deux nombres on doit calculer le biais (décalage) :

$B = 2^{5-1} = 2^4 = 16$

$N1 = (-0,014)_8 = (-0,000001100)_2 = (-0,1100)_2 . 2^{-5}$

$ExpB = -5 + 16 = 11 = (01011)_2$

$N2 = (0,14)_8 = (0,001100)_2 = (0,1100)_2 . 2^{-2}$

$ExpB = -2 + 16 = 14 = (01110)_2$

Donc, on va avoir la représentation suivante pour N1 et N2:

N1 :

| | | |
|---|---|---|
| 1 | 01011 | 1100000000 |

1010111100000000 $(AF00)_{18}$

N2 :

| | | |
|---|---|---|
| 1 | 01110 | 1100000000 |

0011101100000000 $(3B00)_{18}$

**3-** Le calcul N1-N2 :

N2 - N1 = 0,14 – (-0,014) = 0,14 + 0,014

N2 - N1 $= (0,1100)_2 . 2^{-2} + (0,1100)_2 . 2^{-5}$

$= (0,1100)_2 . 2^{-2} + (0,0001100)_2 \ 2^{-2}$

$= (0,1101100)_2 . 2^{-2}$

- Si on fait les calculs avec l'exposant biaisé :

N2 - N1 $= (0,1100)_2 . 2^{14} + (0,1100)_2 . 2^{11}$

$= (0,1100)_2 . 2^{14} + (0,0001100)_2 . 2^{14}$

$= (0,1101100)_2 . 2^{14}$

Exposant biaisé = 14

Exposant réel = Exposant biaisé – Biais

Exposant réel = 14 – 16 = -2

Donc, on trouve le même résultat que la première opération.